Exhibit F

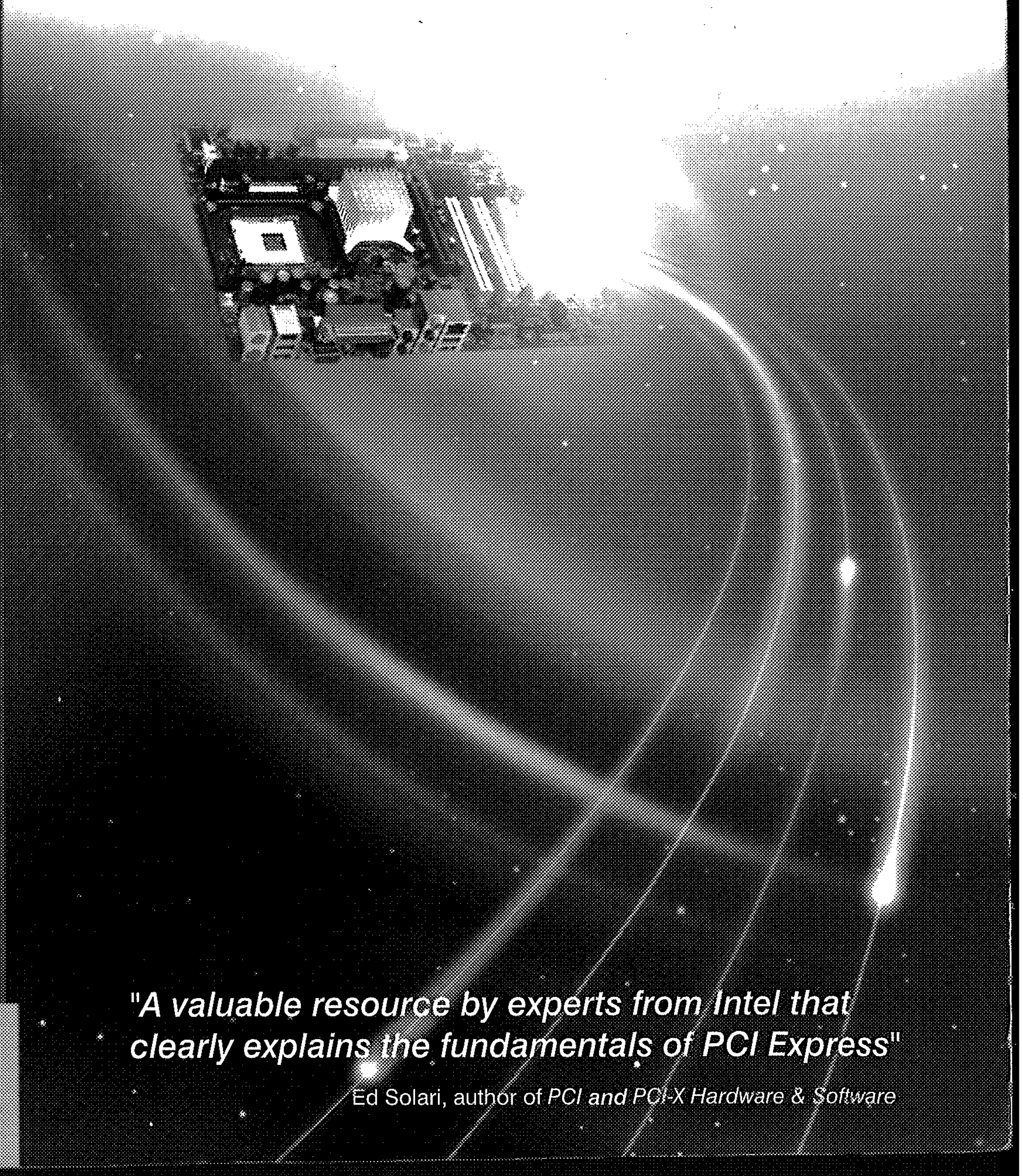
Introduction to PCI Express†

A Hardware and Software Developer's Guide

Adam Wilen, Justin Schade, and Ron Thornburg

INTEL PRESS

Books by Engineers, for Engineers



intel®

"A valuable resource by experts from Intel that clearly explains the fundamentals of PCI Express"

Ed Solari, author of *PCI and PCI-X Hardware & Software*

Introduction to PCI Express†

A Hardware and Software Developer's Guide

Adam H. Wilen
Justin P. Schade
Ron Thornburg

INTEL PRESS

ISBN 0-9702846-9-1

This book is printed on acid-free paper. ♾

Publisher: Richard Bowles
Editor: David J. Clark
Managing Editor: David B. Spencer
Content Manager: Matt Wangler
Text Design: Marianne Phelps
Composition: WASSER Studios
Graphic Art: Donna Lawless (illustrations), Ted Cyrek (cover)

Library of Congress Cataloging in Publication Data:

Printed in the United States of America

10 9 8 7 6 5 4 3 2

Chapter 5 PCI Express Architecture Overview

Always design a thing by considering it in its next larger context—a chair in a room, a room in a house, a house in an environment, an environment in a city plan.

—Eliel Saarinen

This chapter introduces the PCI Express architecture, starting off with a system level view. This addresses the basics of a point-to-point architecture, the various types of devices and the methods for information flow through those devices. Next, the chapter drops down one level to further investigate the transaction types, mainly the types of information that can be exchanged and the methods for doing so. Lastly, the chapter drops down one level further to see how a PCI Express device actually goes about building those transactions. PCI Express uses three transaction build layers, the Transaction Layer, the Data Link Layer and the Physical Layer. These architectural build layers are touched upon in this chapter with more details in Chapters 6 through 8.

System Level Overview

Whereas PCI is a parallel, multi-drop interface, PCI Express is a serial, point-to-point interface. As such, many of the rules and interactions in PCI are no longer directly applicable to PCI Express. For example, devices no longer need to arbitrate for the right to be the bus driver prior to sending out a transaction. A PCI Express device is always the driver for its transmitter pair(s) and is always the target for its receiver pair(s). Since only one device ever resides at the other end of a PCI Express link, only one device can drive each signal and only one device receives that signal.

In Figure 5.1, Device B always drives data out its differential transmitter pair (traces 1 and 2) and always receives data on its differential receiver pair (traces 3 and 4). Device A follows the same rules, but its transmitter and receive pairs are mirrored to Device B. Traces 3 and 4 connect to Device A's transmitter pair (TX), while traces 1 and 2 connect to its receiver pair (RX). This is a very important difference from parallel busses such as PCI; the transmit pair of one device *must* be the receiver pair for the other device. They must be point-to-point, one device to a second device. TX of one is RX of the other and vice versa.

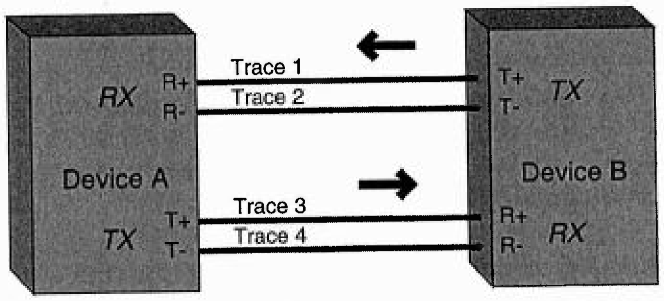


Figure 5.1 Point-to-Point Connection between Two PCI Express Devices

Links and Lanes

The connection between two PCI Express devices is referred to as a *link*. A link consists of a number of lanes, much the same way that a highway consists of a number of driving lanes. With PCI Express, a *lane* is the term used for a single set of differential transmit and receive pairs. A lane contains four signals, a differential pair for unidirectional transmission in both directions (referred to as dual unidirectional). The link shown in Figure 5.1 is a single lane wide. The link shown in Figure 5.2 is 4 lanes wide.

At the device, the collection of transmitter and receiver pairs that are associated with a link is referred to as a *port*. Like links, a device's port can be made up of multiple lanes. A quick aside on terminology: a PCI Express device and its associated port(s) refer to signals as either transmit (TX) or receive (RX) signals. However, when one refers to a link, the signals that make up each lane cannot truly be called "transmitter" or "receiver". No given connection is just a transmitter or a receiver because it is considered both simultaneously. The transmit signal for Device B is the receive signal for Device A.

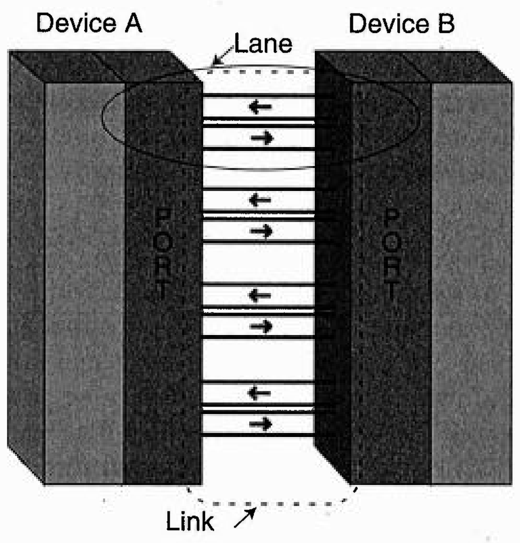


Figure 5.2 Links, Lanes, and Ports

The signaling scheme for PCI Express is tremendously simple. Each lane is just a unidirectional transmit pair and receive pair. There are no separate address and data signals, no control signals like the FRAME#, IRDY# or PME# signals used in PCI, not even a sideband clock sent along with the data. Because of this simplicity and modularity, the architecture can more easily scale into the future, provide additional bandwidth and simplify the adoption of new usage models. However, it also requires the adoption of signaling techniques vastly different from traditional PCI.

Embedded Clocking

PCI Express utilizes 8-bit/10-bit encoding to embed the clock within the data stream being transmitted. At initialization, the two devices determine the fastest signaling rate supported by both devices. The current specification only identifies a single signaling rate, 2.5 gigabits per second (per lane per direction), so that negotiation is pretty simple. Since the transfer rate is determined ahead of time, the only other function of the clock would be for sampling purposes at the receiver. That is where 8-bit/10-bit encoding with an embedded clock comes into play. By transmitting each byte of data as 10 encoded bits, you can control the number of transitions associated with each transmission character—simplifying the sampling procedures on the receiver side. Chapter 8, "Physical Layer Architecture" contains more information on this topic.

Multiple Lanes

You might now be asking yourself, "If the transfer rate is fixed ahead of time at 2.5 gigabits per second per lane per direction, how can this interface scale to meet the needs of high-bandwidth interfaces?" After all, 2.5 gigabits per second per direction is only 250 megabytes per second of actual data that can flow each way (recall that with 8-bit/10-bit encoding, each byte of data is transferred as 10 bits, so you need to divide 2.5 gigabits per second by 10 to get theoretical data transfer rates). A data transfer rate of 250 megabytes per second per direction might be better than traditional PCI, but it certainly is not in the same league as higher bandwidth interfaces like AGP (AGP4x runs at 1 gigabyte per second and AGP8x runs at 2 gigabytes per second total bandwidth). When you add to this the fact that a parallel bus, like PCI or AGP, is substantially more efficient (at the same frequency) than a serial interface like PCI Express, the bandwidth of this new interface seems to be at a disadvantage to some existing platform technologies. Well, that is where PCI Express' scalability comes into play.

Much like lanes can be added to a highway to increase the total traffic throughput, multiple lanes can be used within a PCI Express link to increase the available bandwidth. In order to make its capabilities clear, a link is named for the number of lanes it has. For example, the link shown in Figure 5.2 is called a x4 (read as: "by four") link since it consists of a four lanes. A link with only a single lane, as in Figure 5.1, is called a x1 link. As previously noted, the maximum bandwidth of a x1 is 250 megabytes per second in each direction. Because PCI Express is dual unidirectional, this offers a maximum theoretical bandwidth of 500 megabytes

Chapter 9

Flow Control

My expressway runneth over.

—Victor Ross, spokesman for NY Bureau of Traffic Operations

This chapter goes into the details of the various flow control mechanisms within PCI Express. It begins with a description of the ordering requirements for the various transaction types and then explains key aspects of the credit-based flow control mechanisms. The rest of the chapter then deals with some of the advanced mechanisms that PCI Express uses to manage the traffic within the system, namely virtual channels and traffic classes. Following that, the chapter briefly describes how these mechanisms are used to support isochronous data streams.

Transaction Ordering

The *PCI Express Base Specification* defines several ordering rules to govern which types of transactions are allowed to pass or be passed. Passing occurs when a newer transaction bypasses a previously issued transaction and the device executes the newer transaction first. The ordering rules apply uniformly to all transaction types—memory, I/O, configuration, and messages—but only within a given traffic class. There are no ordering rules between transactions with different traffic classes. It follows that there are no ordering rules between different virtual channels, since a single traffic class cannot be mapped to multiple virtual channels within a given link. Further details on this are located in the section on virtual channels and traffic classes later in this chapter. Please note that the ordering rules for completions are somewhat decoupled from request ordering. Completions use their own ordering rules and do not necessarily follow the same ordering as their associated requests.

Table 9.1 illustrates the various rules for when a transaction must, may, or cannot be allowed to pass a previously issued transaction. An entry of Yes indicates that to avoid deadlock, the subsequent transaction (identified by that row) must be allowed to pass the previous transaction (as identified by that column). An entry of No indicates that there are no specified requirements. The subsequent transaction may pass or be blocked by the previous transaction. An entry of No indicates that the subsequent transaction must not pass the previous transaction.

Table 9.1 Ordering Rules

Row Pass Column?		Posted Request	Non-Posted Request		Completion	
(Row letter or column number in parentheses)		Memory Write or Message Request (1)	Read Request (2)	I/O or Config Write Request (3)	Read Completion (4)	I/O or Config Write Completion (5)
Posted Request	Memory Write or Message Request (A)	a) No b) Y/N	Yes	Yes	a) Y/N b) Yes	a) Y/N b) Yes
Non-Posted Request	Read Request (B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Config Write Request (C)	No	Y/N	Y/N	Y/N	Y/N
Completion	Read Completion (D)	a) No b) Y/N	Yes	Yes	a) Y/N b) No	Y/N
	I/O or Config Write Completion (E)	Y/N	Yes	Yes	Y/N	Y/N

A subsequent memory write or message request interacts with previous transactions as follows. As seen in cell A1, there are two potential ordering rules when dealing with two memory write or message requests. If the relaxed ordering bit (bit 5 of byte 2 in the TLP header) contains a value of 0, then the second transaction is not permitted to bypass the previously submitted request (A1a). If that bit is set to a 1, then the subsequent transaction is permitted to bypass the previous transaction (A1b). A memory write or message request must be allowed to pass read requests (A2) as well as I/O or configuration write requests (A3) in order to avoid deadlock. The ordering rules between memory write or message requests and completion packets depend on the type of PCI Express device. Endpoints, switches, and root complexes may allow memory write or message requests to pass or be blocked by completions (A4a and A5a). PCI Express to PCI or PCI-X bridges, on the other hand, must allow memory write or message requests to pass completions in order to avoid deadlock (A4b and A5b). This scenario only occurs for traffic flowing from the upstream (PCI Express) side of the bridge to the downstream (PCI or PCI-X) side of the bridge.

A subsequent non-posted request (any read request or an I/O or configuration write request) interacts with previous transactions in the following way. As seen in cells B1 and C1, these requests are not allowed to pass previously issued memory write or message requests. Non-posted requests may pass or be blocked by all other transaction types (B2, B3, B4, B5, C2, C3, C4, and C5).

A subsequent read completion interacts with previous transactions as follows. As seen in cell D1, there are two potential ordering rules when determining if a read completion can pass a previously issued memory write or message request. If the relaxed ordering bit (bit 5 of byte 2 in the TLP header) contains a value of 0, then the read completion is not permitted to bypass the previously submitted request (D1a). If that bit is set to a 1, then the read completion may bypass the previously enqueued transaction (D1b). A read completion must be allowed to pass read requests (D2) as well as I/O or configuration write requests (D3) in order to avoid deadlock. Read completions from different read requests are treated in a similar fashion to I/O or configuration write completions. In either case (D4a or D5), the subsequent read completion may pass or be blocked by the previous completion transaction. Recall however, that a single completion may be split up amongst several completion packets. In this scenario, a subsequent read completion packet is not allowed to pass a previously enqueued read completion packet for that same request/completion (D4b). This is done in order to ensure that read completions return in the proper order.

A subsequent I/O or configuration write completion interacts with previous transactions as follows. As seen in cell E1, these completions may pass or be blocked by previously issued memory write or message requests. Like read completions, I/O or configuration write completions must be allowed to pass read and I/O or configuration write requests in order to avoid deadlocks (E2 and E3). Finally, as seen in E4 and E5, I/O and configuration write completions may pass or be blocked by previously issued completions.

Flow Control

PCI Express enacts flow control (FC) mechanisms to prevent receiver buffer overflow and to enable compliance with the ordering rules outlined previously. Flow control is done on a per-link basis, managing the traffic between a device and its link mate. Flow control mechanisms do not manage traffic on an end-to-end basis, as shown in Figure 9.1.

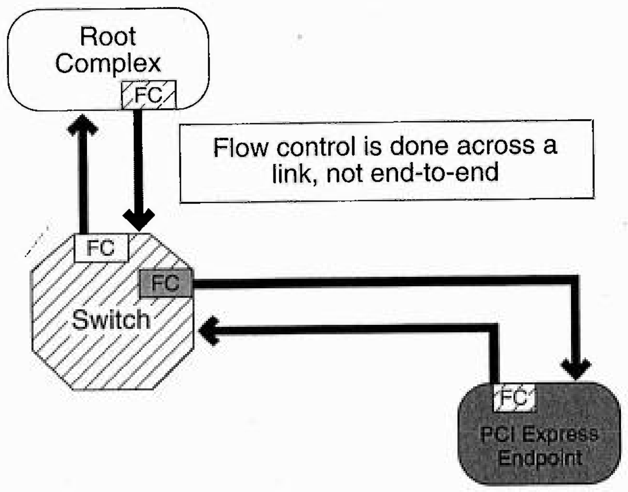


Figure 9.1 Link by Link Flow Control

In the example in Figure 9.1, the root complex issues a request packet destined for the PCI Express endpoint and transmits that packet across the outgoing portion of its link to the switch. The switch then sends that packet across its downstream port to the endpoint. The flow control mechanisms that PCI Express implements, however, are local to each link. The flow control block in the root complex only deals with managing the traffic between the root complex and the switch. The downstream port of the switch and the endpoint then manage the flow control for that packet between the switch and the endpoint. In Figure 9.1, there are no flow control mechanisms in the root complex that track the packet all the way down to the endpoint.

Link mates share flow control information to ensure that no device transmits a packet that its link mate is unable to accept. Each device indicates how many flow control credits it has available for use. If the next packet allocated for transmission exceeds the available credits at the receiver, that packet cannot be transmitted. Within a given link, each virtual channel maintains its own flow control credit pool.

As mentioned in Chapter 7, DLLPs carry flow control details between link mates. These DLLPs may initialize or update the various flow control credit pools used by a link. Though the flow control packets are DLLPs and not TLPs, the actual flow control procedures are a function of the Transaction Layer in cooperation with the Data Link Layer. The Transaction Layer performs flow control accounting for received TLPs and gates outgoing TLPs if they exceed the credits available. The flow control mechanisms are independent of the data integrity mechanisms of the Data Link Layer (that is, the flow control logic does not know if the Data Link Layer was forced to retry a given TLP, and retry cannot be used as a form of flow control).

Flow Control Rules

What is flow control really doing? It helps to ensure that traffic flows through a system in an orderly manner. Just imagine trying to drive down the highway if there were no rules governing how to drive. What would stop somebody else from driving down your side of the road? What would stop somebody in the next lane from pulling directly into your lane? During high traffic times, what if nobody bothered to use their brakes? Cars that tried to occupy the same space at the same time would collide, and cars that had no place to go could be forced to swerve off the road and be lost. Traffic rules help to avoid these sorts of issues on the highway, and flow control rules help avoid these same sorts of issues on a PCI Express link.

First consider a single lane bridge that must service cars in both directions. To get access to the road, a driver must arbitrate for control of the road with other drivers going both the same direction and the opposite direction. This is a good representation of how conventional PCI and PCI-X flow control works. Additionally, once a car gains access to the road, it needs to determine how fast it can go. If there is a lot of traffic already on the road, the driver may need to throttle his or her advancement to keep from colliding with other cars on the road. PCI accomplishes this through signals such as IRDY# and TRDY#.

Now consider that the road is changed into a highway with four lanes in both directions. This highway has a carpool lane that allows carpoolers an easier path to travel during rush hour traffic congestion. There are also fast lanes for swifter moving traffic and slow lanes for big trucks and other slow moving traffic. Drivers can use different lanes in either direction to get to a particular destination. Each driver occupies a lane based upon the type of driver he or she is. Carpoolers take the carpool lane while fast drivers and slow drivers occupy the fast and slow lanes respectively. This highway example represents the PCI Express flow control model. Providing additional lanes of traffic increases the total number of cars or bandwidth that can be supported. This is what is accomplished by adding additional lanes to a PCI Express link. Prioritizing who gets to use the available bandwidth (especially during high traffic times) is what virtual channels and traffic classes add to the picture.

PCI Express does not have the same sideband signals (IRDY#, TRDY#, RBF#, WBF#, and so on) that PCI or AGP have in order to implement this sort of flow control model. Instead, PCI Express uses a flow control credit model. Data Link Layer Packets (DLLPs) are exchanged between link mates indicating how much free space is available for various types of traffic. This information is exchanged at initialization, and then updated throughout the active time of the link. The exchange of this information allows the transmitter to know how much traffic it can allow on to the link, and when the transmitter needs to throttle that traffic to avoid an overflow condition at the receiver.

Flow control differentiates between various types of TLPs and allocates separate credit pools for each type. TLPs are divided up into the following types for flow control purposes: posted request header (PH), posted request data (PD), non-posted request header (NPH), non-posted request data (NPD), completion header (CplH), and completion data (CplD). Posted request credits (Px) apply to message and memory write requests. Non-posted request credits (NPx) apply to IO and configuration write, as well as all read requests. Completion credits (Cplx) apply to the completions associated with a corresponding request.

For the various data credits, the corresponding unit of credit is equal to 16 bytes of data (that is to say that 1 CplD unit equals 16 bytes of completion data). For the various header credits, the corresponding unit of credit is the maximum-size header plus TLP digest. Table 9.2 identifies the credits associated with the various types of traffic.

Table 9.2 TLP Flow Control Credits

TLP	Credits Consumed
Memory, I/O, configuration read request	1 NPH
Memory write request	1 PH + n PD
I/O, configuration write request	1 NPH + 1 NPD (note: size of data written for these TLPs is never more than one aligned DWord)
Message request without data	1 PH
Message request with data	1 PH + n PD
Memory read completion	1 CplH + n CPLD
I/O, configuration read completions	1 CplH + 1 CPLD
I/O, configuration write completions	1 CplH

The *n* units used for the data credits are calculated by rounding up the data length by 16 bytes. For example, a memory read completion with a data length of 10 DWords (40 bytes) uses 1 CplH unit and 3 (40/16 = 2.5, which rounds up to 3) CplD units. Please note that there are no credits and hence no flow control processes for DLLPs. The receiver must process these packets at the rate that they arrive.

Each virtual channel has independent flow control, and thus maintains independent flow control pools (buffers) for PH, PD, NPH, NPD, CplH, and CplD credits. Each device autonomously initializes the flow control for its default virtual channel (VC0). As discussed in Chapter 7, this is done during the DL_Init portion of the Data Link Layer state machine. The initialization procedures for other virtual channels flow control are quite similar to that of VC0, except that VC0 undergoes initialization by default (and before the link is considered active) while other virtual channels undergo initialization after the link is active. Once enabled by software, multiple virtual channels may progress through the various stages of initialization simultaneously. They need not initialize in numeric VC ID order (that is to say, VC1 initializes before VC2 initializes before VC3, and so on) nor does one channel's initialization need to complete before another can begin (aside from VC0, which must be initialized before the link is considered active). Additionally, since VC0 is active prior to the initialization of any other virtual channels, there may already be TLP traffic flowing across that virtual channel. Such traffic has no direct impact on the initialization procedures for other virtual channels. Additional details on virtual channel and flow control initialization are contained later in this chapter.

Flow Control Credits

As discussed previously, there are six types of flow control credits: PH, PD, NPH, NPD, CplH, and CplD. Initialization takes care of providing the initial values for each of these credit types. This is accomplished via the Init_FC1 and Init_FC2 DLLPs. During this initialization, receivers must initially advertise virtual channel credit values equal to or greater than those shown in Table 9.3.

Table 9.3 Minimum Initial Flow Control Credit Advertisements

Credit Type	Minimum Advertisement
PH	1 unit – credit value of 01h
PD	Largest possible setting of the Max_Payload_Size for the component divided by the FC unit size (16 bytes) For example, if the Max_Data_Payload size is 1024 bytes, the smallest permissible initial credit value would be 040h (64 decimal)
NPH	1 unit – credit value of 01h
NPD	1 unit – credit value of 01h
CplH	Switch: 1 unit – credit value of 01h Root complex and endpoint: infinite FC units – initial credit value of all 0s – this is interpreted by the transmitter as infinite, which allows it to never throttle
CplD	Switch: Largest possible setting of the Max_Payload_Size for the component divided by the FC unit size (16 bytes), or the largest read request the unit will ever generate (whichever is smaller) Root complex and endpoint: infinite FC units – initial credit value of all 0s – this is interpreted by the transmitter as infinite, which allows it to never throttle